Synthesis of Software Designs from Requirements

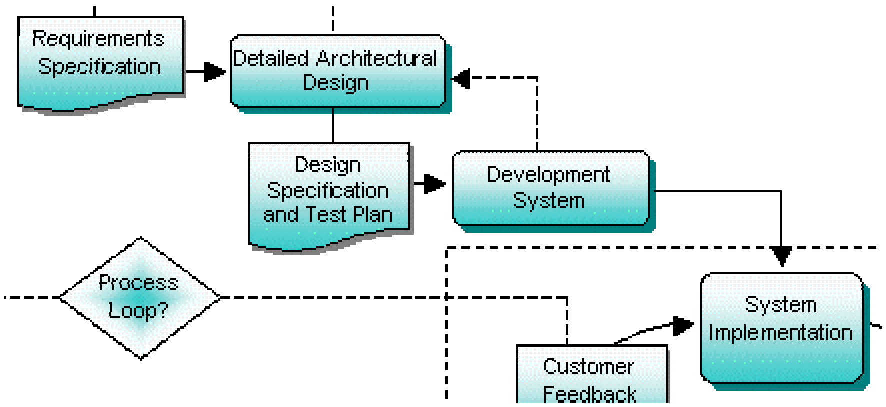
Photo: NASA Ames Research Center

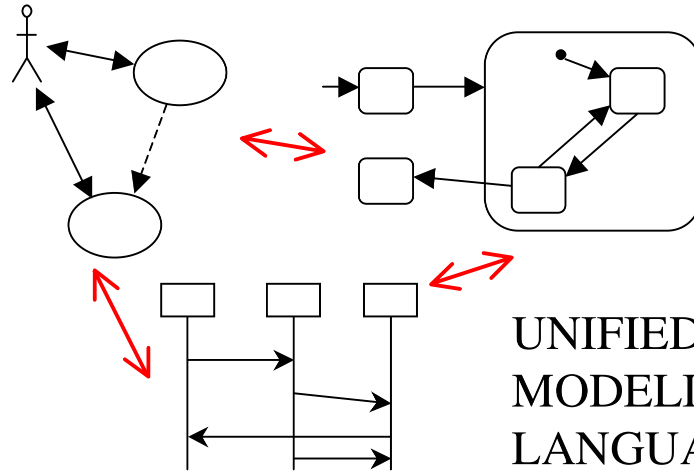
Jon Whittle

QSS Group Inc.

NASA Ames Research Center

Software Modeling: now and future



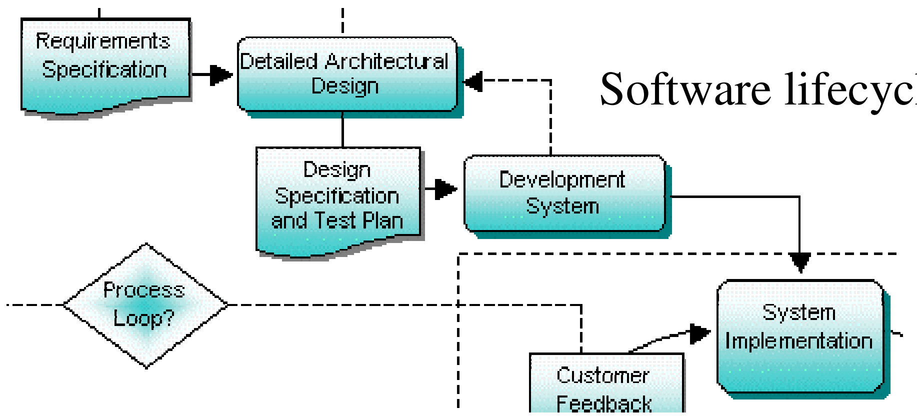


UNIFIED MODELING LANGUAGE

- A **software model** is a **blueprint for software**: essential for project team communication and to ensure architectural soundness
- NASA missions generally follow rigorous software processes that increasingly rely on software modeling, e.g., Mission Data System (JPL), Space Shuttle (United Space Alliance)
- Currently, however, software models serve *merely as documentation* that becomes *obsolete* when **crunch time** hits

Our research goal: develop algorithms and tools that allow software models to be kept in sync with each other

Software Modeling

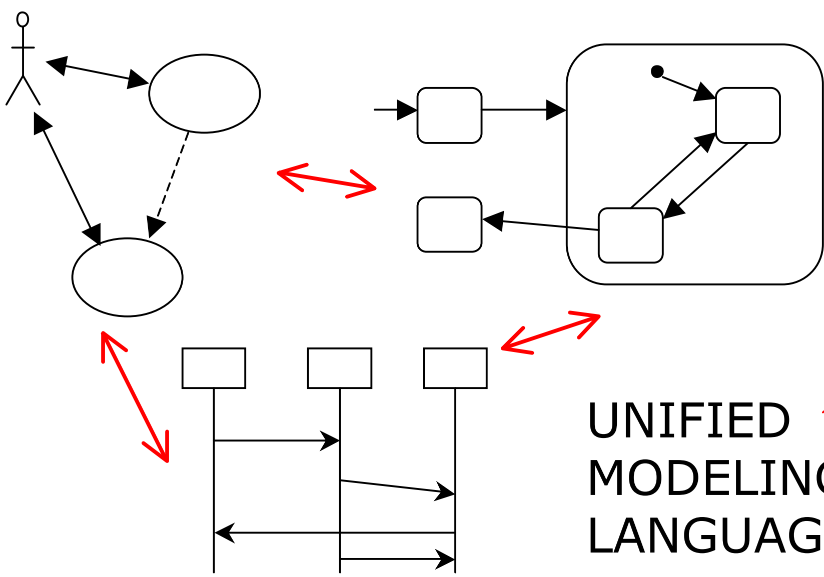


Software lifecycle

Advantages:
- *common language*
- *easy to document code*
- *each stakeholder chooses his own notation*
- *multiple viewpoints*

Problems:
- *loose connection between models*
- *difficult to maintain consistency*
- *redundancy in models*

Research Goal:
algorithms, methodologies and tool support for connecting models in a consistent way



UNIFIED
MODELING
LANGUAGE

- Unifies OO methods
- Modeling language for OO
- Notations + meta-model
- Commercial tools: Rhapsody, Rose etc.
- Used in NASA: MDS, Space Shuttle, ATC

Ames Research Center

Software Modeling with UML

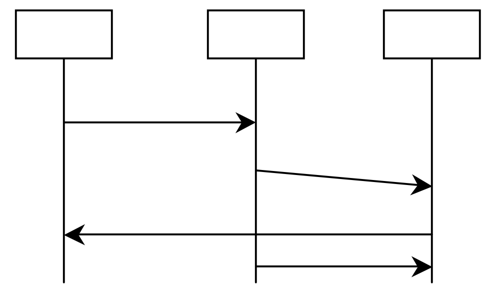
Use cases (scenarios) — Requirements

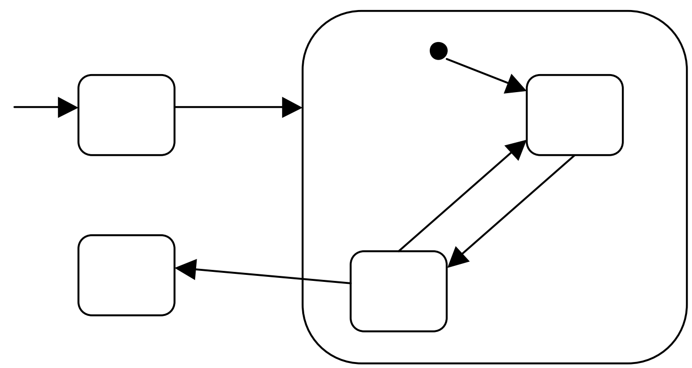
???

Statecharts — Design

Code generators

C++/Java/… — Code

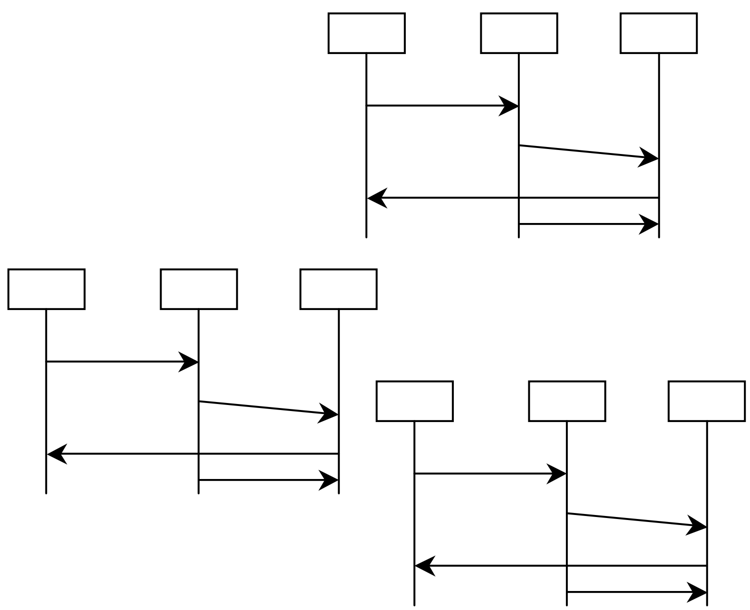




- Unifies OO methods
- Modeling language for OO
- Notations + meta-model
- Commercial tools: Rhapsody, Rose etc.
- Used in NASA: MDS, Space Shuttle, ATC

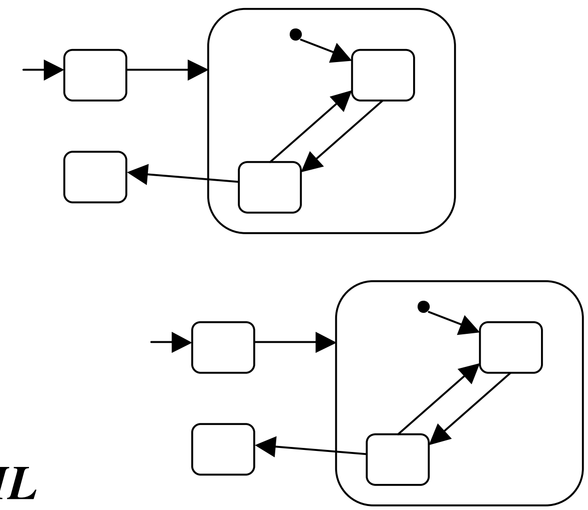
Design Synthesis From Scenarios

Requirement Scenarios (global view)



Statechart synthesis from scenarios (UML sequence diagrams)

Design (local view)



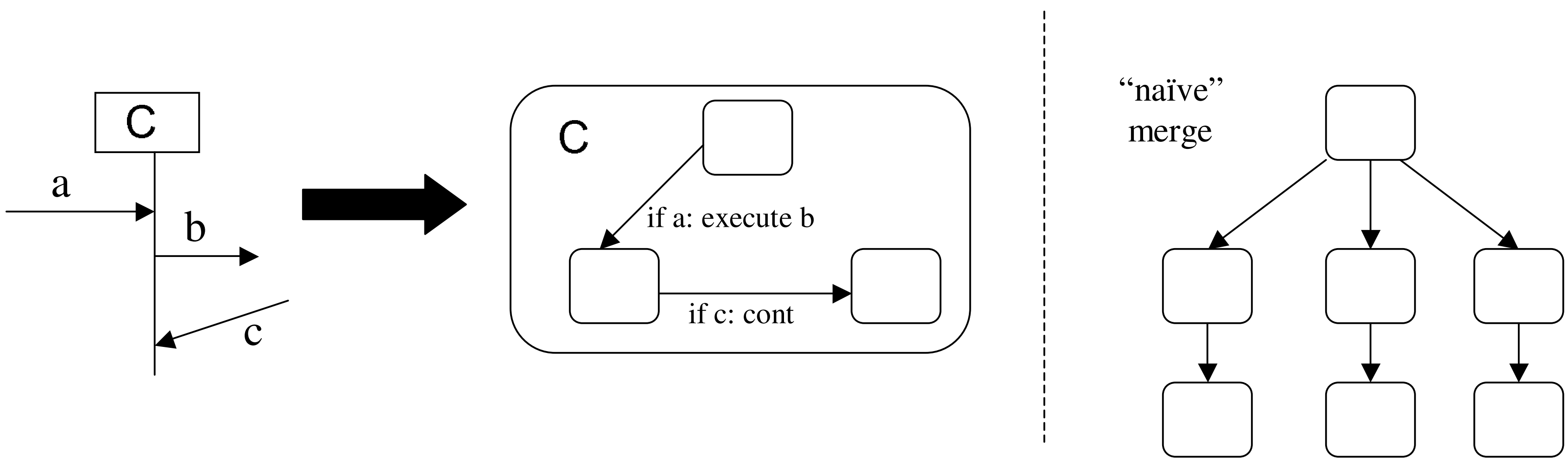
Technical challenge:
- *detect inconsistencies*
- *merge duplicated behaviors*
- *generate readable designs*
- *facilitate iterative refinements*

What is a “good” synthesis machine?

- Related work:
  - Harel’s LSCs
  - Systä et al (MAS)
  - Khriss et al
- Detect inconsistencies & ambiguities
- Merge similar or duplicated behaviors
- Generate *readable* statecharts
- Facilitate iterative refinements

Technology Developed

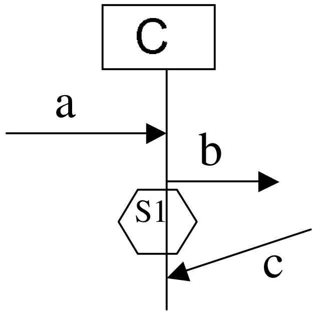
- Developed an algorithm for automatically translating UML sequence diagrams into UML statecharts that embodies the idea of a “good” synthesis engine
- Previous state of the art based around a “naïve” merging:
  - Generate a statechart for each class/component
  - Regard incoming messages as trigger events, outgoing messages as actions
  - Merge each class’s statecharts from all the sequence diagrams “naively”



- ScGen supercedes this using advanced technology to merge states if appropriate, introduce structure into the states automatically and detecting conflicts in the scenarios

Scenarios & Semantic Content

- A "good" synthesis engine requires additional semantic information
    - ➢ Use Explicit Labels or Class Diagram Constraints (OCL)
- Explicit Labels:



- OCL Constraints:

  Context Agent::graspBox(b : Box)

  pre: self.carries->isEmpty

  post: self.carries->includes(b)

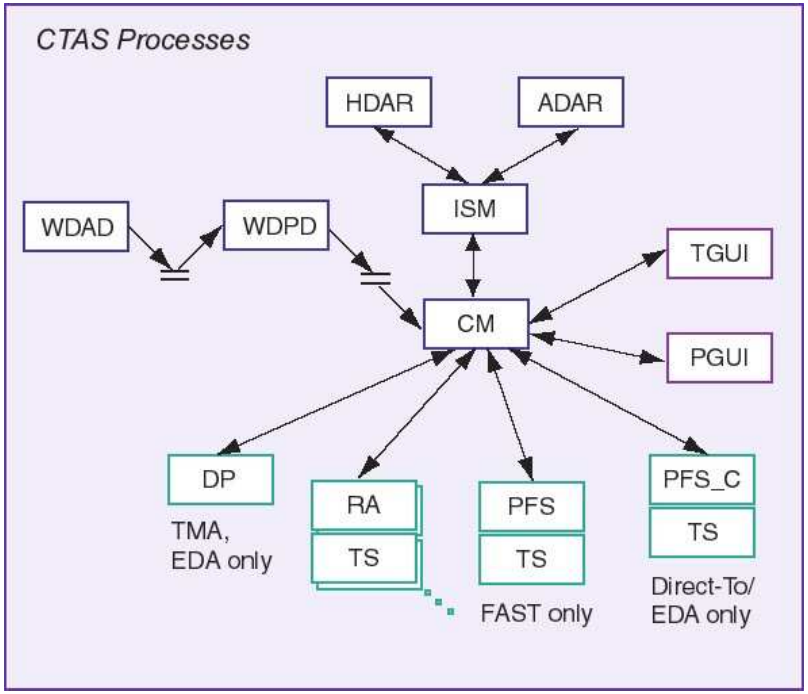
- SD1: graspBox, move

  SD2: graspBox, graspBox, move

- Characterize states using *State Vector :*
    - *SV = <orient', gripper', hands', carries', coordWith', leader' >*
- Use state vector to merge states, detect conflicts and introduce hierarchy:

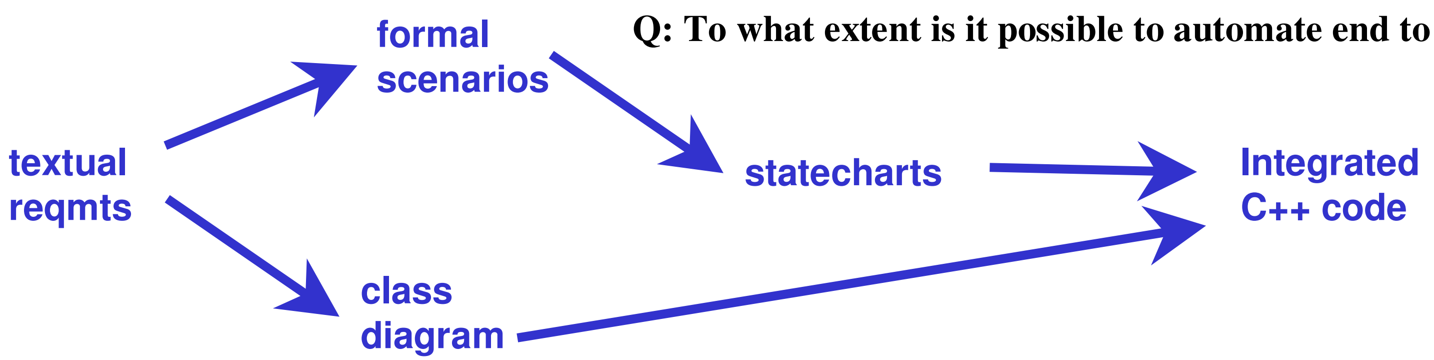
  → via a unify/propagate/conflict detection algorithm

Case study:
Center Tracon Automation System (CTAS)



- Set of tools to increase throughput at airports/ in US airspace
- Already implemented, but constantly evolving
- Weather control logic -distribute weather updates to all clients

Q: To what extent is it possible to automate end to end synthesis?



Requirements

- *"every client that uses weather data should be notified of a weather update and all clients should begin using the updated weather data at the same time"*
- 10 pages of textual requirements, e.g.:

2.8.16. The CM should perform the following actions when the Weather Cycle status is POSTUPDATING and any connected weather-aware client has responded no to the CTAS_USE_NEW_WTHR message:

  - **a) it should set the Weather Cycle status to DONE;**
  - **b) it should remove the Weather Cycle pending state**
  - **c) it should enable the F2 weather control panel "set" button**
  - **d) it should write the new weather forecast information to the cmsim file**
  - **e) it should send CM_CLOSE_CONNECTION messages to all connected weather-aware clients**

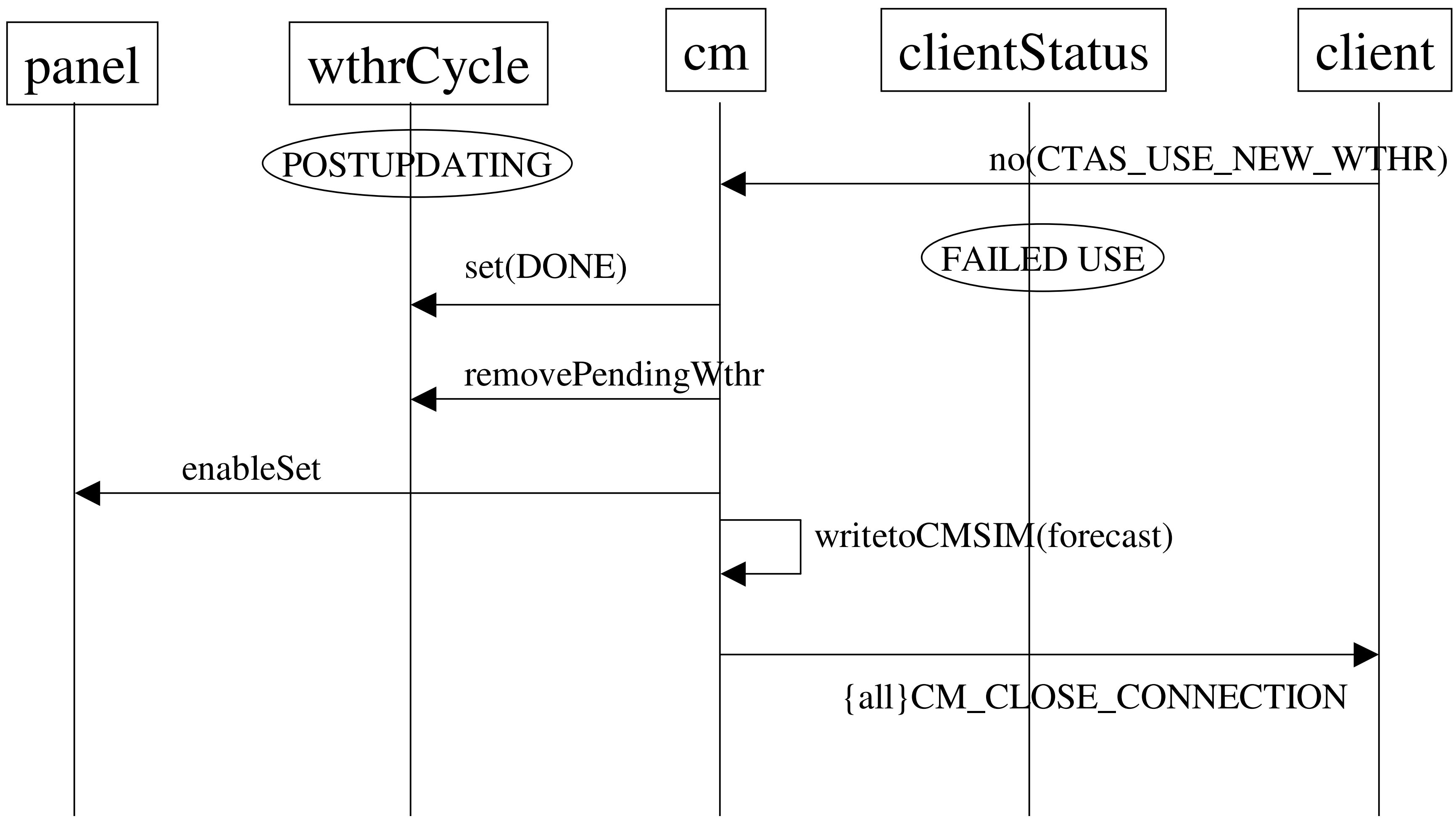
- Generic form:

$$\text{IN(state) and Prec} \Rightarrow \text{Actions}$$

- 46 scenarios + structural information

Formal Scenarios

- Transcribed 46 scenarios into UML sequence diagrams (2-20 messages each):



Statechart and Code Synthesis

- Statecharts for CM synthesized using ScGen algorithm (ICSE2000) extended to deal with state labels.
- Class diagram developed from requirements and populated with method declarations
- Method bodies reverse engineered from existing CTAS code
- C++ code generated using RoseRT

Future Work

- **Design synthesis from scenarios**
  - Support design synthesis with *patterns*
  - Develop algorithms for the “backwards direction”
  - Application to reverse engineering
- **Other models**
  - Algorithms for translating between multiple viewpoints for other UML models: e.g., in developing consistent domain models for state estimation program synthesis

Synthesis of Software Designs from Requirements

Objective:

Increase the effectiveness and efficiency of software modeling within NASA by:

- *developing advanced algorithms that support software modeling*
- *focusing on modeling languages in use within NASA – namely, the Unified Modeling Language (UML)*
- *integrating the algorithms and their implementations into NASA software projects*

Benefits:

- *Increased productivity and reliability in large NASA software projects*
- *Reusability between NASA software projects*

Principal Investigator:

Jon Whittle, QSS Group Inc.
NASA Ames Research Center
+1 650 604 3589
jonathw@email.arc.nasa.gov

Project Researchers:

Jon Whittle and Johann Schumann,
NASA Ames Research Center

Significant Accomplishments

- ***Prototype tool, ScGen**, designed and implemented in Java*
- *Paper on ScGen won a **Best Paper Award** at the International Conference on Software Engineering*
- *Research team have organized a number of **successful, high profile workshops** in this field: at OOPSLA and ICSE conferences*
- *ScGen currently undergoing **application case study** within the CTAS software development process at NASA Ames*
- *Silicon Valley start-up interested in **commercialization***